6. Categorical and Crossed Variables

6.1 Introduction

Many survey variables encode discrete categorical meanings rather than represent underlying continuous values. Although the CREATE step initially interprets input data as real or real with missing, the CATEGORICAL or CAT statement may be used to indicate the categorical interpretation of one or more variables. The first half of this chapter concerns categorical variables.

Once one or more categorical variables have been defined, the CROSS statement may be used to cross-classify a real or real with missing variable by a categorical variable, creating a crossed real variable. A categorical variable may be crossed by another, creating a crossed categorical variable. Crossed real variables may be crossed by additional categorical variables, allowing the construction of higher dimensional crossed real variables. Similarly, crossed categorical variables may be crossed multiple times.

Class variables, described in the next chapter, are an alternative to the use of crossed variables. The considerations are:

- Crossing provides a quick way to produce cross-classified data that DISPLAY produces in a compact form, as Section 6.7 will illustrate.

- CLASS variables are better suited to allow further treatment of the resulting arrays by features of the TRANSFORM step and manipulation in the DISPLAY step to form a variety of marginals and subarrays of a multi-dimensional array.

The discussion of categorical variables in the first half of the chapter should interest most VPLX users, but a cursory initial reading of the material on crossing may be an effective approach. This chapter assumes a reading of Chapters 3 and 4.

Summary of this chapter:

- Section 6.2 describes the CATEGORICAL or CAT statement.

- Section 6.3 discusses the treatment of categorical variables by the DISPLAY step.

- Section 6.4 provides an example.

- Section 6.5 details how categorical variables are treated in the CREATE step with respect to features such as COPY, IF blocks, *etc*.

- Section 6.6 describes the CROSS statement, which produces crossed real and crossed categorical variables.

- Section 6.7 describes the treatment of crossed variables in the DISPLAY step.

- Section 6.8 provides an example.

- Section 6.9 details how crossed variables are treated in the CREATE step with respect to features such as COPY, IF blocks, *etc*.

6.2 The CATEGORICAL or CAT Statement

6.2.1 CATEGORICAL (CAT). All variables listed on the INPUT list are read as real variables, although they often represent categorical rather than continuous data. The purpose of the CATEGORICAL, or CAT, statement serves to regroup real variables into discrete categories. VPLX will save separate counts or weighted tallies for each category. The basic syntax of this statement is

```
CAT  vlist (range1 / range2 /...) [' label1' ['label2'[ ...]]]
```

where *vlist* contains a list of one or more variables, the ranges represent values to be assigned to each of the categories, and, optionally, labels of up to 24 characters for each category are provided. The "/"'s in the syntax separate the different categorical levels of variable. For example,

```
CATEGORICAL   SEX  (1/2) 'Male' 'Female'
```

reclassifies the variable SEX as categorical by placing values of 1 into the first category and 2 into the second.

Section 3.6.3 describes range specifications. VPLX does not edit the range specifications for overlap. The range specifications are examined in the order specified, and the observation will be classified into the first level in which the specification is satisfied. For example, for

```
CATEGORICAL   SEX  (1 /1-2) 'Male' 'Female'
```

VPLX will classify the value 1 into the first level without noting that the second range is satisfied as well.

Values falling outside of all the specified ranges will be excluded from the categorical variable but will not cause an error.

The CAT or CATEGORICAL statement may be used to classify several variables at once, for example, a series of opinion questions might be treated by

```
cat  q103 - q117 q121 q123 (1/2/3,4/res) 'Yes', 'No',
     'Don''t know or refused','Missing/no answer'
```

Note that this example uses `res`, described by Section 3.6.3, to include all remaining values in the last level.

It is also possible to create new variables while retaining the old. This syntax takes the form:

```
CAT  vlist1 INTO vlist2 (range1 / range2 ...) ['label1'
      ['label2'[...]]]
```

where ***vlist1*** and ***vlist2*** must have the same length, that is, the same number of variables. New variables appearing in ***vlist2*** become defined by this statement. For example,

```
CAT TOTAL_EARN INTO EARN_CAT (0/1-9999/10000-24999/25000-49999
     /50000-High)
```

`EARN_CAT` becomes defined as a categorical variable with five categories while `TOTAL_EARN` is retained as a real variable.

When labels are omitted, VPLX will use the specified ranges as labels, that is, "0", "1-9999" etc., in the preceding example.

Single-Level Categorical Variables: In some cases, it is advantageous to specify categorical variables with only one level. For example, one may want only the estimated total for some characteristic, such as persons receiving social security income. Section 6.3 describes the somewhat different treatment by the DISPLAY step of single-level categorical variables.

6.2.2 Labels. The preceding example establishes labels for each category. Note that labels can be separated by commas or blanks. Apostrophes between adjacent labels should not touch, however; like FORTRAN, VPLX translates a pair of adjacent apostrophes into a single apostrophe in the label, as in `'Bachelor''s Degrees'` . Individual labels should be started

and completed on the same command line. A command line with an odd number of apostrophes is an error.

6.2.3 Storage of Categorical Variables: To obtain the weighted counts for each level of a categorical variable, VPLX reserves a cell in the cumulative totals for each category. At the observation level, however, VPLX simply records in a single cell the number of the level to which the observation belongs. If the observation did not fall into any of the specified categories, a 0 is held instead. Exhibit 6.1 illustrates the storage requirements for categorical variables, using as a specific instance a categorical variable with 3 levels:

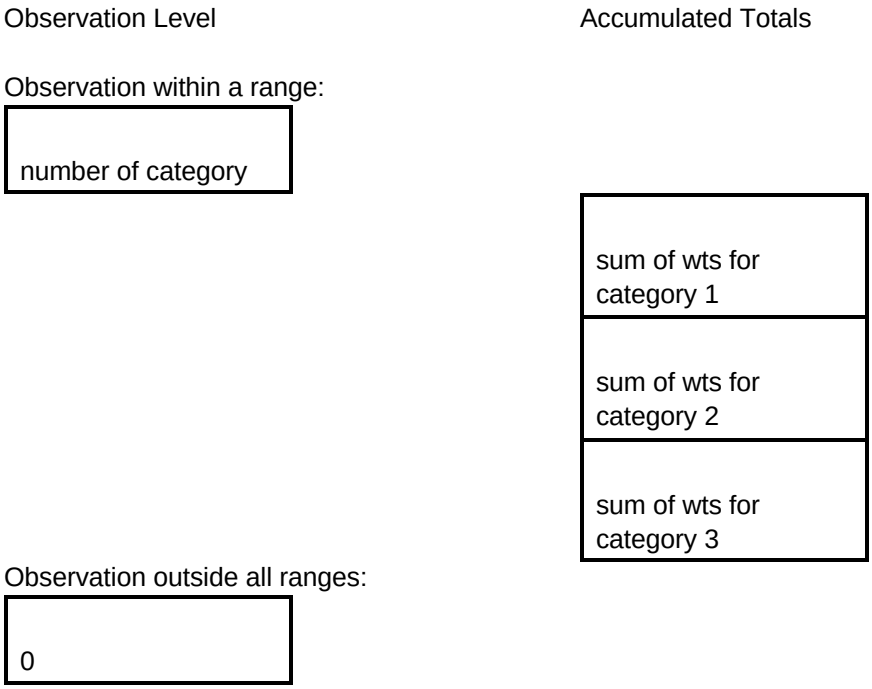

Exhibit 6.1 Representation at the observation and cumulative level for a categorical variable. The left-hand side shows the observation-level treatment of a categorical variable. The right-hand side shows the representation of cumulative sums over observations, both for the overall total and for each replicate. At the observation level, a value is held indicating into which category a variable should be summed. In this example, the right-hand side shows the categorical variable to have 3 levels, so that VPLX stores on the left-hand side a 1, 2, or 3 to represent the category to which the observation belongs. When the observation does not fall into any of the specified intervals, a 0 is stored on the left-hand side and the observation does not contribute to the totals on the right.

6.3 Treatment of Categorical Variables in the DISPLAY STEP

The DISPLAY step treats categorical variables substantially differently from real and real with missing variables. In many cases the treatment of single-level categorical variables is different from other cases. The following standard functions are available for categorical variables:
MEAN or **MEANS** - This function is interpreted as PERCENT for a categorical variable .

N - This function provides the sum of weighted counts of included observations in the defined categories. The sum omits observations falling outside of the valid ranges. For single-level categorical variables, however, the value is the number of included observations in the data set, or the block to which the categorical variable belongs (*1*)

PERCENT or **PERCENTS** - This function is the default for a categorical variable. For a categorical variable with 2 or more categories, the percentages are computed with respect to the weighted count of the included cases in the categorization. If a categorical variable has a single level, the denominator is the weighted number of cases. (More specifically, the weighted number of cases for the block.)

PERCENT1 - If this function is applied to a categorical or crossed categorical data, then statistics are computed in the same manner as PERCENT, but the final category is not shown. This function is particularly helpful, for example, for dichotomous variables, since PERCENT will produce two lines with estimates summing to 100 and identical standard errors, while PERCENT1 produces only the first percentage with its appropriate standard error.

PROPORTION - This function is like PERCENT, but values are shown as proportions.

PROPORTION1 - This function is identical to PERCENT1 except for showing proportions instead of percentages.

TOTAL or **TOTALS** - If it is applied to categorical variable, then the function gives the weighted counts for each category.

TOTAL1 - This function is like PERCENT1, except that totals are shown.

6.4 An Example of CATEGORICAL.

The following example is based on an elaboration of example 4 of Section 2.3.

```
comment  EXAM18

comment  This example modifies EXAM4.CRD, based on the simple
         jackknife.  To illustrate some features of CATEGORICAL
         new categorical variables, rooms_cat, and persons_cat,
         will be added.
         The transformation step will be skipped.

create  in = example1.dat  out = example1.vpl

input    rooms persons cluster

      3 variables are specified

format   (3f2.0)

cat     rooms into rooms_cat (1-5/6-high) '1-5 rooms' '6 or more rooms'

cat     persons into persons_cat (1-3/4-high) '1-3 persons'
                    '4 or more persons'

print   rooms persons rooms_cat persons_cat / options nprint = 3
    *** PRINT request    1

labels  rooms 'Number of rooms' persons 'Persons'
        rooms_cat 'Number of rooms' persons_cat 'Number of persons'

    (Simple) jackknife replication assumed

    Size of block   1  =              7

    Total size of tally matrix =      7

    Unnamed scratch file opened on unit 13

    Unnamed scratch file opened on unit 14

**** End of CREATE specification/beginning of execution
PRINT request    1
     rooms                  5.000000
     persons                7.000000
     rooms_cat              1.                  Categorical
     persons_cat            2.                  Categorical
PRINT request    1
     rooms                  6.000000
     persons                8.000000
     rooms_cat              2.                  Categorical
     persons_cat            2.                  Categorical
PRINT request    1
     rooms                  5.000000
     persons                2.000000
     rooms_cat              1.                  Categorical
     persons_cat            1.                  Categorical

    End of primary input file after obs #       6

display

options  totals
```

```
list     rooms  persons rooms_cat persons_cat  total1 (rooms_cat persons_cat)

options  means

list     rooms  persons rooms_cat persons_cat
          proportion (rooms_cat persons_cat )
          percent1 (rooms_cat persons_cat)
          proportion1 (rooms_cat persons_cat )
          n (rooms_cat persons_cat)
```

			Estimate	Standard error
Number of rooms	:	TOTAL	36.0000	4.0988
Persons	:	TOTAL	24.0000	7.0993
Number of rooms	:	TOTAL		
1-5 rooms			3.0000	1.3416
6 or more rooms			3.0000	1.3416
Number of persons	:	TOTAL		
1-3 persons			3.0000	1.3416
4 or more persons			3.0000	1.3416
Number of rooms	:	TOTAL	3.0000	1.3416
Number of persons	:	TOTAL	3.0000	1.3416

			Estimate	Standard error
Number of rooms	:	MEAN	6.0000	.6831
Persons	:	MEAN	4.0000	1.1832
Number of rooms	:	PERCENTS		
1-5 rooms			50.0000	22.3607
6 or more rooms			50.0000	22.3607
Number of persons	:	PERCENTS		
1-3 persons			50.0000	22.3607
4 or more persons			50.0000	22.3607
Number of rooms	:	PROPORTION		
1-5 rooms			.5000	.2236
6 or more rooms			.5000	.2236
Number of persons	:	PROPORTION		
1-3 persons			.5000	.2236
4 or more persons			.5000	.2236
Number of rooms	:	PERCENTS	50.0000	22.3607
Number of persons	:	PERCENTS	50.0000	22.3607
Number of rooms	:	PROPORTION	.5000	.2236
Number of persons	:	PROPORTION	.5000	.2236

```
Number of rooms         :  WEIGHTED N              6.0000              .0000

Number of persons       :  WEIGHTED N              6.0000              .0000
```

Exhibit 6.2 An example illustrating CATEGORICAL in the CREATE step. Two categorical variables are defined in the CREATE step. The results of the PRINT statement show how the categorical variables hold the levels of the categorical variables. The DISPLAY illustrates functions TOTAL, TOTAL1, PERCENT, PROPORTION, PERCENT1, PROPORTION1, and N applied to 2 categorical variables. Notice that when a categorical variable has only 2 levels, DISPLAY prints a single line for TOTAL1, PERCENT1, and PROPORTION1, omitting the label of the first level. The programming note below describes how the clarity of the DISPLAY can be improved by modifying the variable label to identify the first level.

Programming Note for TOTAL1, PERCENT1, and PROPORTION1. Exhibit 6.2 illustrates that TOTAL1, PERCENT1, and PROPORTION1 display a single line for a 2-level categorical variable by omitting the label of the first level. In many applications, the meaning will be obvious, particularly with "Yes/No" items in which the first level is "Yes." Modifying the variable label may be necessary to maintain clarity. For example,

```
labels  rooms_cat '1-5 Rooms' persons_cat '1-3 Persons'
```

6.5 Treatment of Categorical Variables in the CREATE Step

6.5.1 COPY. If a categorical variable is copied into a target variable, the target assumes all values and attributes of the categorical variable, including variable and level labels.

6.5.2 IF Blocks. A categorical variable may be compared to a range:

```
IF   vname ( range ) THEN
```

As Exhibit 6.1 illustrates, however, the value stored for a categorical variable is the number of the level to which it belongs, or 0 if it fell outside the available categories. The original value of the categorical variable is, in effect, lost in carrying out the CAT statement. VPLX will implement an IF statement involving a comparison of a categorical variable to a range, but prints a warning message. Such comparisons are an easy source of misunderstanding. For example,

```
cat   age (0-14/15-19/20-34/35-49/50-64/65-high)
if    age ( 65 - high ) then
```

The condition will never be satisfied, since, after the CAT statement, the possible values for `age` will be 0-6. Instead,

```
cat   age (0-14/15-19/20-34/35-49/50-64/65-high)
if    age ( 6 ) then
```

correctly begins an IF block for persons age 65 and over.

CATEGORICAL or CAT statements cannot be placed within an IF block. For problems in which it might appear attractive to do so, the alternative is to use IF blocks and other techniques for real variables to produce a real value that can be used in a single CAT statement.

6.6 Forming Cross-Classifications: CROSS

6.6.1 CROSS. It is also possible to obtain specific cross-classifications of one or more variables with a categorical variable through the CROSS statement. The available forms are:

```
CROSS  vlist1 BY vname

CROSS  vlist1 BY vlist2

CROSS  vlist1 BY vname INTO vlist2

CROSS  vname BY vlist1 INTO vlist2

CROSS  vlist1 BY vlist2 INTO vlist3
```

Note that BY is included in each of these forms, and the variable or variables that immediately follow it must be categorical (not including, however, variables following INTO, if it is present). The length of *vlist2*, which appears in all but the first version, and the length of *vlist3* in the last version must agree with the length of *vlist1*.

There are two basic types of crossed variables: crossed real and crossed categorical. Crossing a real, real with missing, or crossed real variable with a categorical variable produces a crossed real variable. Crossing a categorical or crossed categorical variable by a categorical variable produces a crossed categorical variable. Multiple crossings are possible, with no specific restriction on the number of crossings except that imposed by the available storage. Multiple crossings are built up one variable at a time, that is, with multiple CROSS statements, each specifying a separate variable after BY.

As an example,

```
CAT   SEX  (1/2) 'Male' 'Female'
CAT TOTAL_EARN INTO EARN_CAT (0/1-9999/10000-24999/
     25000-49999/50000-High)
cross total_earn earn_cat by sex into t_earn_sex earn_c_sex
```

Assuming that `total_earn` and `sex` are both real before these statements, the last statement creates two new variables, `t_earn_sex` and `earn_c_sex`, of types crossed real and crossed categorical, respectively. Later, `t_earn_sex` could be used in the DISPLAY step to show total or mean earnings by sex, and `earn_c_sex` would be the basis for a categorical income cross-classification and percentage distributions by sex.

The series,

```
cross income by sex
cross income by race
cross income by age
cross income by education
```

where `sex`, `race`, `age`, and `education` are previously defined categorical variables, builds a single cross-classification of income by sex by race by age by education. Continuing the series in this manner for several more variables will exceed available storage in a relatively short time. If what is wanted is a series of tables with income cross-classified by each of the other variables, in turn, then the correct approach is to employ INTO:

```
cross income by sex    into income_sex
cross income by race   into income_race
cross income by age    into income_age
cross income by education  into income_ed
```

or more succinctly:

```
cross income by sex race age education into income_sex
      income_race income_age income_ed
```

The current variable names and variable and level labels of the components of the cross-classification are saved with the crossed variable. Consequently, it is advantageous to declare variable labels (Section 3.15) before CROSS statements that use the variables, otherwise the crossed variables will only store the default labels (the variable names).

6.6.2 Storage of Crossed Variables. Crossed categorical and crossed real are stored in different manners. Exhibit 6.3 illustrates the storage of a crossed categorical variable at the observation

and total level. The exhibit shows specifically how a variable resulting from crossing a 2-level categorical variable by a 3-level categorical variable is stored.

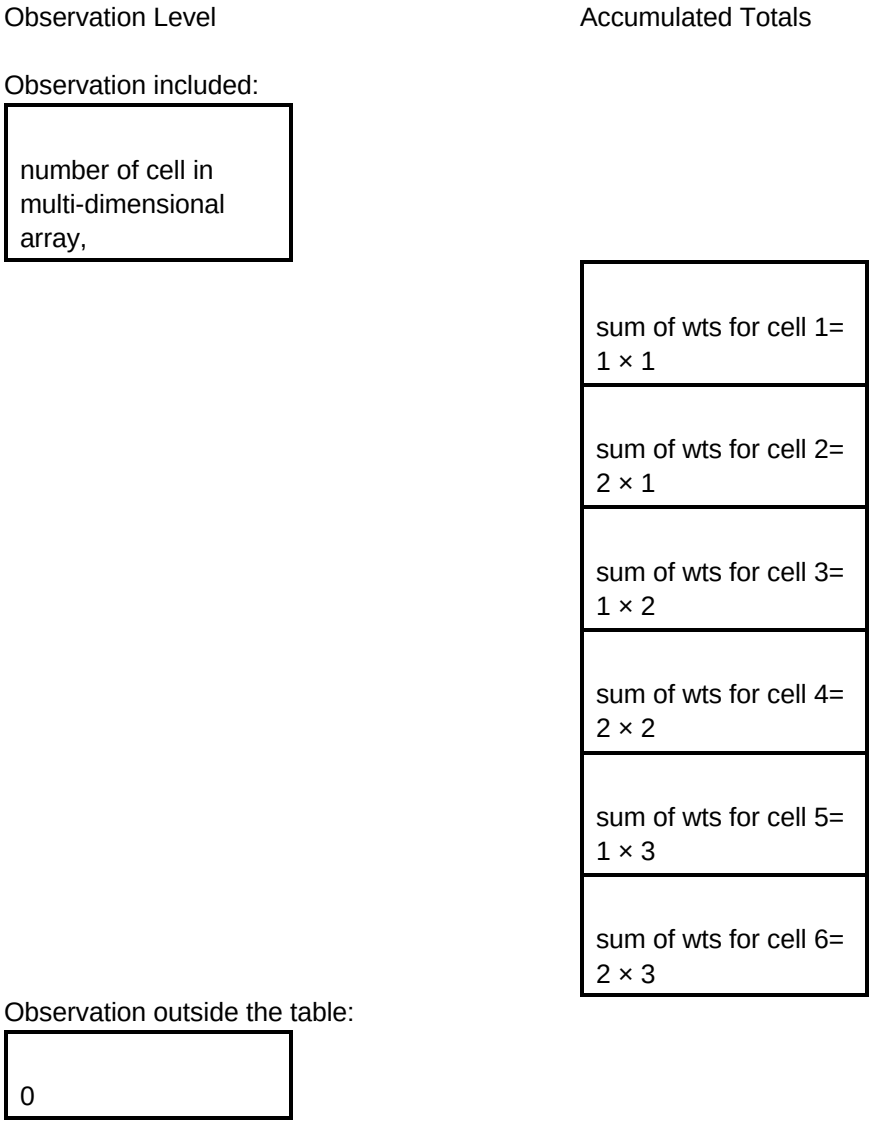

Exhibit 6.3 Representation at the observation and cumulative level for a crossed categorical variable. The specific example is for a 2-level categorical variable crossed by a 3-level one. The left-hand side shows the observation-level treatment of a categorical variable. The right-hand side shows the representation of cumulative sums over observations, both for the overall total and for each replicate. At the observation level, a value is held indicating into which cell an observation belongs. In this example, there are 6 possible cells, so the left-hand side may store 1-6 for included observations. When the observation does not fall into any of the specified intervals, a 0 is stored on the left-hand side and the observation does not contribute to the totals on the right.

6.12

Representation of a crossed real variable requires retention of both the real value and the location of the cell to which the observation belongs. The following example is for a real variable crossed twice: first by a 2-level categorical variable, then by a 3-level categorical variable:

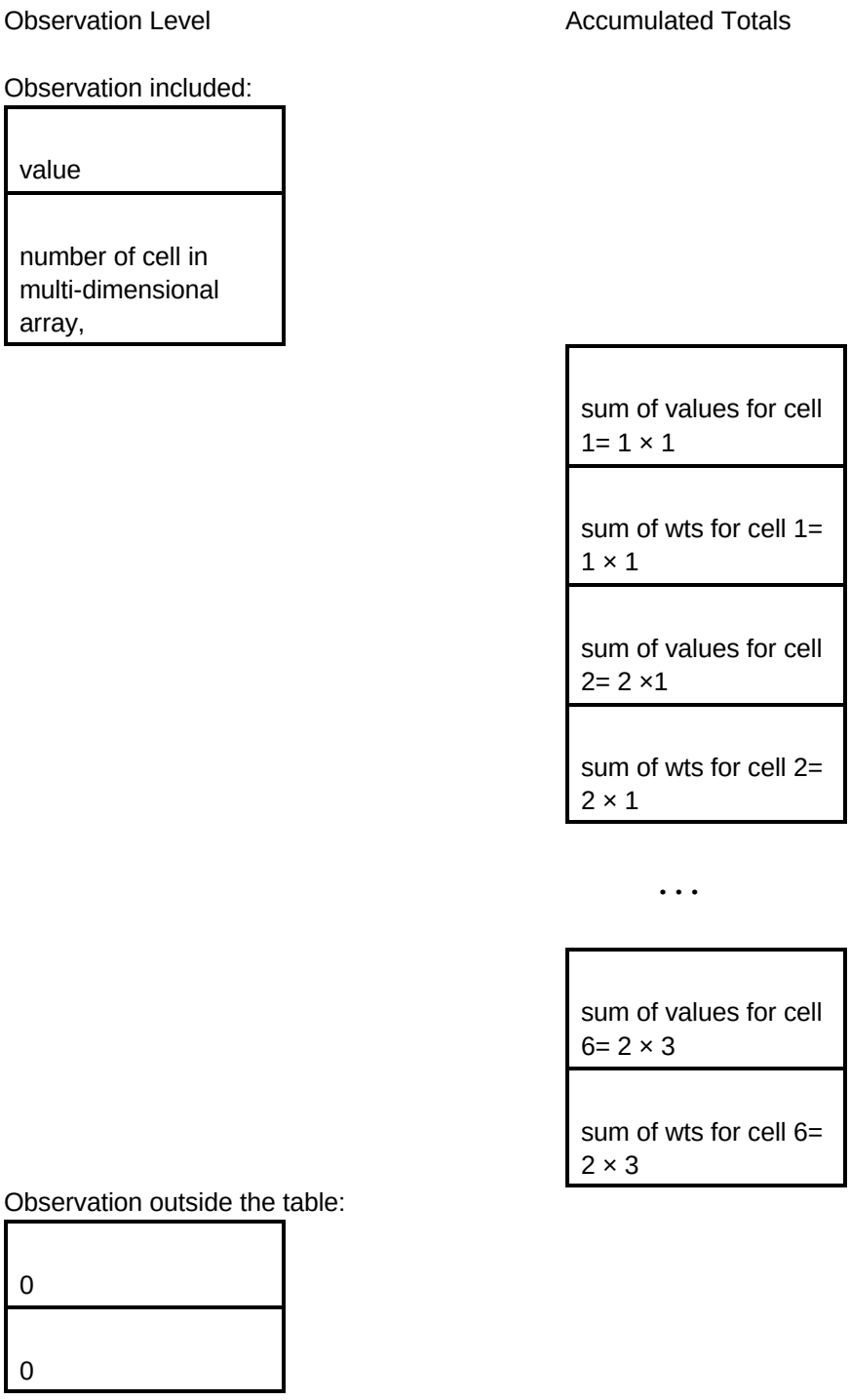

Exhibit 6.4 Representation at the observation and cumulative level for a crossed real variable. The specific example is for a real or real with missing variable crossed by a 2-level categorical variable and then by a 3-level one. The left-hand side shows the observation-level treatment of a crossed real variable. The right-hand side shows the representation of cumulative sums over observations, both for the overall total and for each replicate. At the observation level, the real value is stored, along with a number indicating into which cell the observation belongs. In this example, there are 6 possible table cells. When the observation does not fall into any of the specified intervals, 0's are stored on the left-hand side and the observation does not contribute to the totals on the right. In this example, 12 double precision numbers are required to hold the totals for the values and weighted counts on the right-hand side.

6.7 Treatment of Crossed Variables in the DISPLAY STEP

6.7.1 Crossed Categorical Variables: The DISPLAY step presumes a special role for the first categorical variable in a crossed categorical variable. In effect, the first categorical variable is treated as "dependent," and many of the functions of the display step operate specifically on this variable. As an example,

```
CAT   SEX  (1/2) 'Male' 'Female'
CAT TOTAL_EARN INTO EARN_CAT (0/1-9999/10000-24999/
     25000-49999/50000-High)
cross earn_cat by sex into earn_c_sex
```

In this case, `earn_cat` is the first categorical variable in the crossed variable `earn_c_sex.` Even if `earn_c_sex` is further crossed by other categorical variables, `earn_cat` remains the first categorical variable in the cross.

In general, the DISPLAY step treats the first categorical variable in a crossed variable in a manner similar to a regular categorical variable, but provides information cross-classified by the other categorical variables making up the crossed variable. The following standard functions are available for crossed categorical variables:

MEAN or **MEANS** - This function is interpreted as PERCENT for a crossed categorical variable.

N - This function provides the sum of weighted counts of included observations in the defined categories. The sum is over the levels of the first categorical variable, omitting observations falling outside of the valid ranges. For example, for `earn_c_sex, N(earn_c_sex)` displays 2 counts: the numbers of males and females included in the table.

PERCENT or **PERCENTS** - This function is the default for a crossed categorical variable. The percentages pertain to the first categorical variable in the cross, computed with respect to the

weighted count of the included cases in the categorization. For example, `PERCENT(earn_c_sex)` displays the percentage distribution for earnings for males and females separately.

PERCENT1 - This function produces estimates identical to PERCENT, but DISPLAY does not show the last category for the first categorical variable. For example, `PERCENT1(earn_c_sex)` displays the percentage distribution for earnings for males and females separately, for the first 4 of the 5 levels of earnings. This function is particularly helpful, for example, for when the first categorical variable is dichotomous..

PERCENT2 - This function is provided for crossed categorical variables only. Percentages are computed over the first two categorical variables, instead of just the first as in PERCENT1. An analogous PROPORTION2 is not available.

PROPORTION - This function is like PERCENT, but values are shown as proportions.

PROPORTION1 - This function is identical to PERCENT1 except for showing proportions instead of percentages.

TOTAL or **TOTALS** - This function gives the weighted counts for each category.

TOTAL1 - This function is like PERCENT1, except that totals are shown.

6.7.2 Crossed Real Variables. The DISPLAY step computes functions of the real value crossed by levels of the categorical variable or variables composing the crossing. The functions parallel those for real or real with missing variables.

MEAN or **MEANS** - This function is the default for crossed categorical variables. The mean is computed for each cell of the crossing.

N - This function provides the sum of weighted counts of included observations in the defined categories. The results shown are the denominators used to compute the means.

PERCENT or **PERCENTS** - This function is interpreted as MEAN for a crossed real variable.

TOTAL or **TOTALS** - This function shows the weighted sums of the real value in each cell of the cross classification. In other words, the function displays the numerators used by MEANS.

6.8 An Example of Crossed Real and Categorical Variables

The following example is based on `exam11.crd`.

```
comment  EXAM19

comment  This example modifies EXAM11 and illustrates use of CROSS

create  in = exampl11.dat  out = exampl11.vpl

input    rooms persons cluster tenure

      4 variables are specified

format   (4f2.0)

comment  The input data set contains
 5 7 1 2
 6 8 2 2
 5 2 3 1
 4 1 4 2
 8 4 5 1
 8 2 6 1

cat      tenure  (2/1,3) 'Renter' 'Owner/other'

cross    rooms by tenure into roomsc

cat      rooms into rooms_cat (1-5/6-high) '1-5 rooms' '6 or more rooms'

cat      persons into persons_cat (1-3/4-high) '1-3 persons'
                    '4 or more persons'

cross    rooms_cat by persons_cat into rooms_per

cross    rooms_cat by tenure

cross    rooms_per by tenure

labels   rooms 'Orig # of rooms' roomsc '# Rooms crossed by tenure'
         rooms_cat 'Intervals of rooms by tenure' rooms_per
          'Rooms by persons by tenure'


    (Simple) jackknife replication assumed

    Size of block   1  =            23

    Total size of tally matrix =     23

    Unnamed scratch file opened on unit 13

    Unnamed scratch file opened on unit 14

**** End of CREATE specification/beginning of execution
```

End of primary input file after obs # 6

```
display

option   ndecimal=2 totals

list    rooms  roomsc rooms_cat rooms_per
        n ( rooms roomsc rooms_cat rooms_per )
        mean (rooms roomsc) percent ( rooms_cat rooms_per )
        percent1 (rooms_cat rooms_per )
```

	Estimate	Standard error
Orig # of rooms : TOTAL	36.00	4.10
Crossed values of Orig # of rooms : TOTAL		
tenure		
Renter	15.00	6.88
Owner/other	21.00	9.77
Crossed values of Intervals of rooms by te: TOTAL		
tenure : Renter		
1-5 rooms	2.00	1.26
6 or more rooms	1.00	1.00
tenure : Owner/other		
1-5 rooms	1.00	1.00
6 or more rooms	2.00	1.26
Crossed values of Intervals of rooms by te: TOTAL		
tenure : Renter		
persons_cat : 1-3 persons		
1-5 rooms	1.00	1.00
6 or more rooms	.00	.00
persons_cat : 4 or more persons		
1-5 rooms	1.00	1.00
6 or more rooms	1.00	1.00
tenure : Owner/other		
persons_cat : 1-3 persons		
1-5 rooms	1.00	1.00
6 or more rooms	1.00	1.00
persons_cat : 4 or more persons		
1-5 rooms	.00	.00
6 or more rooms	1.00	1.00
Orig # of rooms : WEIGHTED N	6.00	.00
Crossed values of Orig # of rooms : WEIGHTED N		
tenure		
Renter	3.00	1.34
Owner/other	3.00	1.34
Crossed values of Intervals of rooms by te: WEIGHTED N		
tenure		
Renter	3.00	1.34
Owner/other	3.00	1.34
Crossed values of Intervals of rooms by te: WEIGHTED N		
tenure : Renter		

```
   persons_cat
      1-3 persons                                  1.00              1.00
      4 or more persons                            2.00              1.26
   tenure                  :  Owner/other
   persons_cat
      1-3 persons                                  2.00              1.26
      4 or more persons                            1.00              1.00

Orig # of rooms         :  MEAN                     6.00               .68

Crossed values of Orig # of rooms         : MEAN
   tenure
      Renter                                       5.00               .65
      Owner/other                                  7.00              1.12

Crossed values of Intervals of rooms by te: PERCENTS
   tenure                  :  Renter
      1-5 rooms                                   66.67             37.27
      6 or more rooms                             33.33             37.27
   tenure                  :  Owner/other
      1-5 rooms                                   33.33             37.27
      6 or more rooms                             66.67             37.27

Crossed values of Intervals of rooms by te: PERCENTS
   tenure                  :  Renter
   persons_cat             :  1-3 persons
      1-5 rooms                                  100.00               .00*
      6 or more rooms                               .00               .00*
   persons_cat             :  4 or more persons
      1-5 rooms                                   50.00             64.55
      6 or more rooms                             50.00             64.55
   tenure                  :  Owner/other
   persons_cat             :  1-3 persons
      1-5 rooms                                   50.00             64.55
      6 or more rooms                             50.00             64.55
   persons_cat             :  4 or more persons
      1-5 rooms                                     .00               .00*
      6 or more rooms                            100.00               .00*

Crossed values of Intervals of rooms by te: PERCENTS
   tenure
      Renter                                      66.67             37.27
      Owner/other                                 33.33             37.27

Crossed values of Intervals of rooms by te: PERCENTS
   tenure                  :  Renter
   persons_cat
      1-3 persons                                100.00               .00*
      4 or more persons                           50.00             64.55
   tenure                  :  Owner/other
   persons_cat
      1-3 persons                                 50.00             64.55
      4 or more persons                             .00               .00*
```

Exhibit 6.5 An example illustrating crossed real variables. Three categorical variables are defined in the CREATE step. From these, CROSS statements define ROOMSC as a

crossed real variable, and ROOMS_CAT and ROOMS_PER as crossed categorical. The DISPLAY illustrates functions TOTAL, N, MEAN, PERCENT, and PERCENT1.

6.9 Treatment of Crossed Variables in the CREATE Step

6.9.1 COPY. If either a crossed real or categorical variable is copied into a target variable, the target assumes all values and attributes of the crossed variable, including variable and level labels.

6.9.2 IF Blocks. Crossed categorical variables may not be used in IF statements.

A crossed real variable may be compared to a range:

```
IF   vname ( range ) THEN
```

The rules are similar to real with missing variables. The condition will not be satisfied if the observation does not fall into a cell of the crossed table. Generally, however, it is preferable to avoid using crossed real variables in IF statements.

CROSS statements cannot be placed within an IF block. This restriction is an example of the general rule that VPLX will not allow the type or attributes of variables to change within IF blocks. CROSS always changes the type or at least the crossed dimensions of one or more variables, and VPLX is not designed to accommodate such conditional changes.

NOTES

1. Blocking is described in Chapter 8. In other words, for a single-level categorical variable, the function N provides the base of the percent, just as it does for other categorical variables.